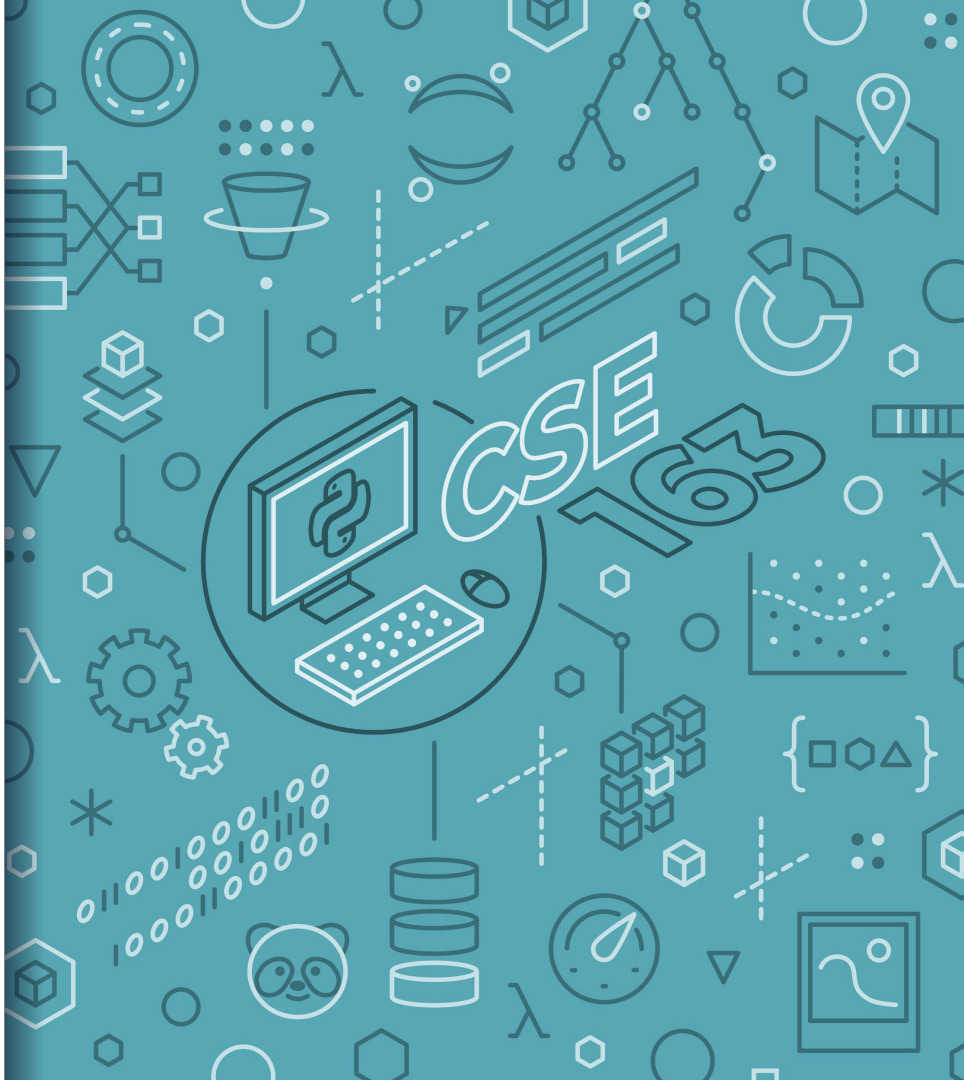
CSE 163

Numpy

Dylan Jergens



Images and Sound

```
array([[ 84,   0,   0, ...,  31,  54,   0],
       [101,   0,   0, ...,   0,  32,   0],
       [  0,  89,   0, ...,  32,   0,   0],
       ...,
       [116,   0,   0, ...,   0,  83,   0],
       [  0,   0, 105, ...,  86,  97,  89],
       [103, 119, 124, ...,   0,   0,   0]])
```

```
array([ 0.97,  1.94,  2.91,  3.88,  4.85,  5.82,  6.79,  7.76,  8.73,
        9.7 , 10.67, 11.64, 12.61, 13.58, 14.55, 15.52, 16.49, 17.46,
       18.43, 19.4 , 20.37, 21.34, 22.31, 23.28, 24.25, 25.22, 26.19,
       27.16, 28.13])
```

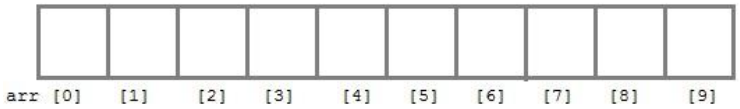
Vectors & Matrices

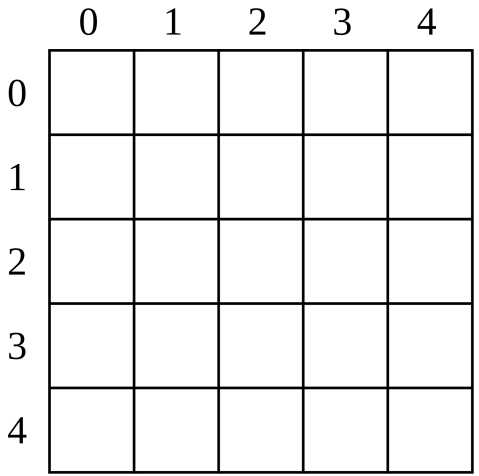
Vectors/Arrays

- One-dimensional arrays.
- Look and behave very similar to Python lists.

Matrices

- Multi dimensional arrays.
- You can think of them as a number of vectors stacked on top of each other.



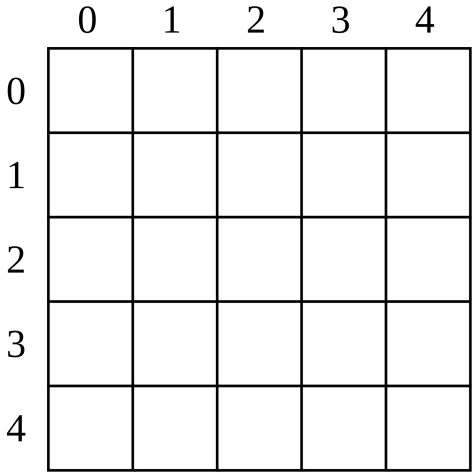


Shape

Numpy arrays have a notion of shape.

- Shape is defined as a tuple (M, N), where M is the number of rows and N is the number of columns.

Shape: (1, 10)



Shape: (5, 5)

Reshaping

Sometimes when we are doing linear algebra, shape is important in the calculations.

Numpy provides a function called **reshape** that allows us to change the shape of an array without change the data inside.

```
v = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
v = v.reshape((3, 3)) # (3 rows, 3 columns)
```

The resulting array:

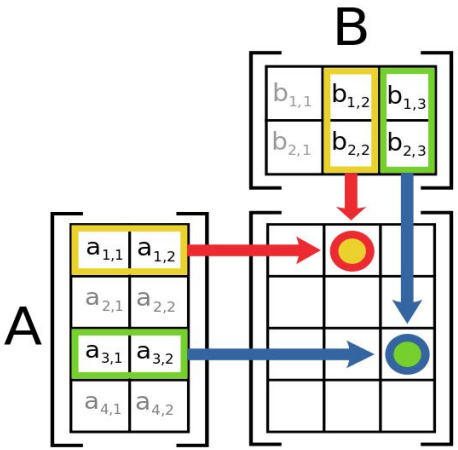
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Multiplication

Numpy also allows us to multiply matrices. This can be super helpful when doing linear algebra calculations.

Matrix A has shape (4, 2)

Matrix B has shape (2, 3)



The result of the multiplication is a new matrix C, with shape (4, 2), where

$$c_{ij} = a_{i1}b_{1j} + ... + a_{im}b_{mj} = \sum_{k=1}^{m} a_{ik}b_{kj}$$

Shape Matters

The result of A x B is a new matrix with shape (4, 3)

When multiplying two matrices, the shape of the two matrices matters!

$$(n \times d) \cdot (d \times k) = (n \times k)$$

The inner dimensions must match, and a matrix with the outer dimensions is produced.

Element-wise Operations

Numpy provides built-in functionality to perform arithmetic operations on vectors and matrices.

```
v = np.array([1, 2, 3])

v + 2   # [3, 4, 5]

v - 1   # [0, 1, 2]

v * 3   # [3, 6, 9]

v / 2   # [0.5, 1., 1.5]

v // 2  # [0, 1, 1]
```

Broadcasting

A set of rules for applying operations to arrays of different sizes.

The Rules of Broadcasting

1. If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded on it's left side.
2. If the shape of two arrays does not match on any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape.
3. If in any dimension the sizes disagree and neither is equal to 1, an error is raised.

Example 1

```
M.shape = (2, 3)
v.shape = (3,)
```

By rule 1, we see that v has fewer dimensions so we pad it on the left with ones.

```
M.shape = (2, 3)
v.shape = (1, 3)
```

By rule 2, we see that the first dimension disagrees, so we stretch the dimensions to match.

```
M.shape = (2, 3)
v.shape = (2, 3)
```

```
v:  [[0, 1, 2],
     [0, 1, 2]]
```

Poll Everywhere

Think

1.5 minutes

pollev.com/cse163

```
np.arange(3).reshape((3, 1)) + np.arange(3)
```

The Rules of Broadcasting

1. If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded on it's left side.
2. If the shape of two arrays does not match on any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape.
3. If in any dimension the sizes disagree and neither is equal to 1, an error is raised.

Poll Everywhere

Pair

2 minutes

pollev.com/cse163

```
np.arange(3).reshape((3, 1)) + np.arange(3)
```

The Rules of Broadcasting

1. If the two arrays differ in their number of dimensions, the shape of the one with fewer dimensions is padded on it's left side.
2. If the shape of two arrays does not match on any dimension, the array with shape equal to 1 in that dimension is stretched to match the other shape.
3. If in any dimension the sizes disagree and neither is equal to 1, an error is raised.

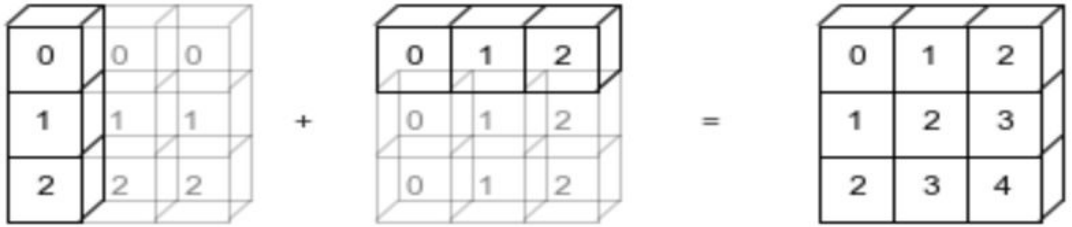
What Happened?

```
a.shape = (3, 1)
b.shape = (3,)
```

```
a.shape = (3, 1)
b.shape = (1, 3)
```

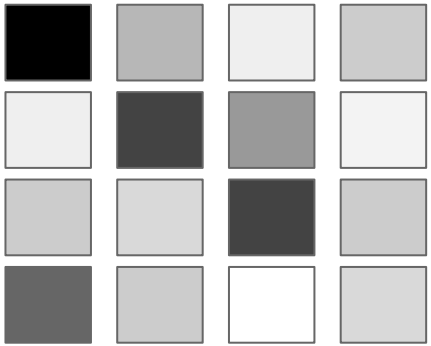
```
a.shape = (3, 3)
b.shape = (3, 3)
```

$\texttt{np.arange(3).reshape((3,1)) + np.arange(3)}$



Images as Matrices

Grey-scale images can be represented as matrices.



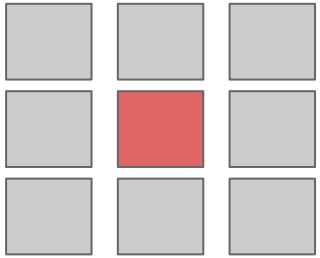
Grey-scale: 255

Grey-scale: 0

```
array([[ 84,   0,   0, ...,  31,  54,   0],
       [101,   0,   0, ...,   0,  32,   0],
       [  0,  89,   0, ...,  32,   0,   0],
       ...,
       [116,   0,   0, ...,   0,  83,   0],
       [  0,   0, 105, ...,  86,  97,  89],
       [103, 119, 124, ...,   0,   0,   0]])
```

Image Smoothing

How can we smooth the image?



We can take the average of the surrounding pixels, and update the current pixel.